Model-driven Information Flow Security for Component-Based Systems

Najah Ben Said, Takoua Abdellatif, Saddek Bensalem, Marius Bozga

Model-driven Information Flow Security for Component-Based Systems*

Najah Ben Said[1], Takoua Abdellatif[2], Saddek Bensalem[1], and Marius Bozga[1]

[1] UJF-Grenoble 1 / CNRS, VERIMAG UMR 5104, Grenoble, F-38041, France
[2] Sousse University, ESSTHS, Hammam Sousse, Tunisia

Abstract. This paper proposes a formal framework for studying information flow security in component-based systems. The security policy is defined and verified from the early steps of the system design. Two kinds of non-interference properties are formally introduced and for both of them, sufficient conditions that ensures and simplifies the automated verification are proposed. The verification is compositional, first locally, by checking the behavior of every atomic component and then globally, by checking the inter-components communication and coordination. The potential benefits are illustrated on a concrete case study about constructing secure heterogeneous distributed systems.

Keywords: component-based systems, information flow security, non-interference, unwinding conditions, automated verification.

1 Introduction

The amount and complexity of nowadays conceived systems and software knows a continuous increase. Information protection and secure information flow between these systems is paramount and represent a great design challenge. Model driven security (MDS) [BDL06] is an innovative approach that tend to solve system-level security issues by providing an advanced modeling process representing security requirements at a high level of abstraction. Indeed, MDS guarantees separation of concerns between functional and security requirements, from early phases of the system development till final implementation.

Information flow security can be ensured using various mechanisms. Amongst the first approaches considered, ones find access control policies [SSM98,Kuh98], that allow protecting data confidentiality by limiting access to data to be read or modified only by authorized users. Unfortunately, these mechanisms have been proven incomplete and limited since only by preventing the direct access to data, indirect (implicit) information flows are still possible given rise to the so called covert channels [SQSL05]. As an alternative, non-interference has been

* The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement ICT-318772 (D-MILS).

studied as a global property to characterize and to develop techniques ensuring information flow security. Initially defined by Goguen and Meseguer [GM82], non-interference ensures that the system's secret information does not affect its public behavior.

In this work, we adapt the MDS approach to develop a component-based framework, named *secBIP*, that guarantees automated verification and implementation of secure information flow systems with respect to specific definition of non-interference. In general, component-based frameworks allow the construction of complex systems by composition of atomic components with communication and coordination operators. That is, systems are obtained from unitary atomic components that can be independently deployed and composed with other components. Component-based frameworks are usually well adopted for managing key issues for functional design including heterogeneity of components, distribution aspects, performance issues, etc. Nonetheless, the use of component-based frameworks is also beneficial for establishing information flow security. Particularly, the explicit system architecture allows tracking easily intra and inter-components information flow.

The *secBIP* framework is built as an extension of the *BIP* [BBS06,BBB$^+$11] framework encompassing information flow security. *secBIP* allows to create systems that are secure by construction if certain local conditions hold for composed components. The *secBIP* extension includes specific annotations for classification of both data and interactions. Thanks to the explicit use of composition operators in *BIP*, the information flow is easily tracked within models and security requirements can be established in a compositional manner, first locally, by checking the behavior of atomic components and then globally, by checking the communication and coordination inter-components.

Information flow security has been traditionally studied separately for language-based models [SS01,SV98] (see also the survey [SM03]) and trace-based models [McC88,McL94,ZL97,Man00]. While the former mostly focus on verification of data-flow security properties in programming languages, the latter is treating security in event-based systems. In *secBIP*, we achieve a useful combination between both aspects, data-flow and event-flow security, in a single semantics model. We introduce and distinguish two types of non-interference, respectively *event non-interference* and *data non-interference*. For events, non-interference states that the observation of public events should not allow to deduce any information about the occurrence of secret events. For data, it states that there is no leakage of secret data into public ones.

The paper is structured as follows. Section 2 recalls the main concepts of the component-based framework adopted in this work. In section 3, we formally introduce the security extension and we provide the two associated definitions of non-interference, respectively for data flows and event flows. Next, in section 4 we formally establish non-interference based on unwinding relations and we provide sufficient conditions that facilitate its automatic verification. In section 5, we provide a use-case as illustrative example. Section 6 discusses the related

work and section 7 concludes and presents some lines for future work. All the proofs of technical results are given in the appendix.

2 Component-Based Design

The *secBIP* framework is built as an extension of the *BIP* framework introduced in [BBS06]. *BIP* stands for *Behavior*, *Interaction* and *Priority*, that is, the three layers used for the definition of components and their composition in this framework. *BIP* allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Such components are transition systems enriched with data. Transitions are used to move from a source to a destination location. Each time a transition is taken, component data (variables) may be assigned new values, computed by user-defined functions (in C). Atomic components are composed by layered application of interactions and priorities. Interactions express synchronization constraints and do the transfer of data between the interacting components. Priorities are used to filter amongst possible interactions and to steer system evolution so as to meet performance requirements e.g., to express scheduling policies.

In this section, we briefly recall the key concepts of *BIP* which are further relevant for dealing with information flow security. In particular, we give a formal definition of atomic components and their composition through multiparty interactions. Priorities are not considered in this work.

2.1 Atomic Components

Definition 1 (atomic component). *An atomic component B is a tuple (L, X, P, T) where L is a set of locations, X is a set of variables, P is a set of ports and $T \subseteq L \times P \times L$ is a set of port labelled transitions. For every port $p \in P$, we denote by X_p the subset of variables exported and available for interaction through p. For every transition $\tau \in T$, we denote by g_τ its guard, that is, a boolean expression defined on X and by f_τ its update function, that is, a parallel assignment $\{x := e_\tau^x\}_{x \in X}$ to variables of X.*

Figure 1 provides an example of an atomic component. It contains two control locations l_1 and l_2 and two ports p_1 and p_2. The transition labeled with p_1 can take place only if the guard $(0 < x)$ is true. When the transition takes place, the variable y is recalculated as some function of x.

Let $\mathcal{D}$ be the data domain of variables. Given a set of variables Y, we call valuation on Y any function $\mathbf{y} : Y \to \mathcal{D}$ mapping variables to data. We denote by $\mathbf{Y}$ the set of all valuations defined on Y.

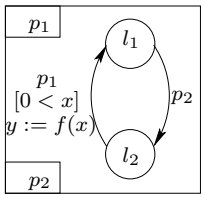

Fig. 1. Atomic Component in *BIP*

Definition 2 (atomic component semantics). *The semantics of an atomic component $B = (L, X, P, T)$ is defined as the labelled transition system* $\text{LTS}(B) = (Q_B, \Sigma_B, \underset{B}{\longrightarrow})$ *where the set of states $Q_B = L \times \mathbf{X}$, the set of labels is $\Sigma_B = P \times \mathbf{X}$ and the set of labelled transitions $\underset{B}{\longrightarrow}$ is defined by the rule:*

$$\textsc{Atom}\ \frac{\tau = \ell \xrightarrow{p} \ell' \in T \qquad \mathbf{x}''_p \in \mathbf{X}_p \qquad g_\tau(\mathbf{x}) \qquad \mathbf{x}' = f_\tau(\mathbf{x}[X_p \leftarrow \mathbf{x}''_p])}{(\ell, \mathbf{x}) \xrightarrow[B]{p(\mathbf{x}''_p)} (\ell', \mathbf{x}')}$$

That is, $(\ell', \mathbf{x}')$ is a successor of $(\ell, \mathbf{x})$ labelled by $p(\mathbf{x}''_p)$ iff (1) $\tau = \ell \xrightarrow{p} \ell'$ is a transition of T, (2) the guard g_τ holds on the current valuation $\mathbf{x}$, (3) $\mathbf{x}''_p$ is a valuation of exported variables X_p and (4) $\mathbf{x}' = f_\tau(\mathbf{x}[X_p \leftarrow \mathbf{x}''_p])$ meaning that, the new valuation $\mathbf{x}'$ is obtained by applying f_τ on $\mathbf{x}$ previously modified according to $\mathbf{x}''_p$. Whenever a p-labelled successor exist in a state, we say that p is *enabled* in that state.

2.2 Composite Components

Composite components are obtained by composing an existing set of atomic components $\{B_i = (L_i, X_i, P_i, T_i)\}_{i=1,n}$ trough specific composition operators. We consider that atomic components have pairwise disjoint sets of states, ports, and variables i.e., for any two $i \neq j$ from $\{1..n\}$, we have $L_i \cap L_j = \emptyset$, $P_i \cap P_j = \emptyset$, and $X_i \cap X_j = \emptyset$. We denote $P = \bigcup_{i=1}^{n} P_i$ the set of all the ports, $L = \bigcup_{i=1}^{n} L_i$ the set of all locations, and $X = \bigcup_{i=1}^{n} X_i$ the set of all variables.

Definition 3 (interaction). *An* interaction *a between atomic components is a triple (P_a, G_a, F_a), where $P_a \subseteq P$ is a set of ports, G_a is a guard, and F_a is an update function. By definition, P_a uses at most one port of every component, that is, $|P_i \cap P_a| \leq 1$ for all $i \in \{1..n\}$. Therefore, we simply denote $P_a = \{p_i\}_{i \in I}$, where $I \subseteq \{1..n\}$ contains the indices of the components involved in a and for all $i \in I, p_i \in P_i$. G_a and F_a are both defined on the variables exported by the ports in P_a (i.e., $\bigcup_{p \in P_a} X_p$).*

Definition 4 (composite component). *A composite component $C = \gamma(B_1, \ldots, B_n)$ is obtained by applying a set of interactions γ to a set of atomic components $B_1, \ldots B_n$.*

Figure 2 presents a classical *Producer-Buffer-Consumer* example modeled in *BIP*. It consists of three atomic components, namely *Producer*, *Buffer* and *Consumer*. The *Buffer* is a shared memory placeholder, which is accessible by both the *Producer* and the *Consumer*. It holds into the local variable x the number of items available. The *Buffer* interacts with the *Producer* (res. *Consumer*) on the *put* (resp. *get*) interaction. On the *put* interaction, an item is added to the *Buffer* and x is incremented. On the *get* interaction, the *Consumer* removes an item from the *Buffer*, if at least one exists (the guard $[x \geq 1]$), and x is decremented.

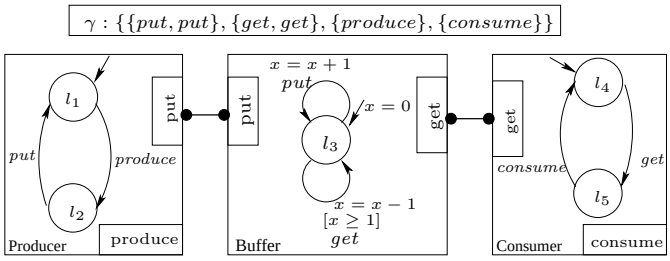

Fig. 2. *BIP* model of the *Producer-Buffer-Consumer* example

Finally, the transitions labeled *produce* and *consume* do not require synchronization - they are executed alone (on singleton port interactions) by the respective components.

Definition 5 (composite component semantics). *Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component. Let $B_i = (L_i, X_i, P_i, T_i)$ and $\text{LTS}(B_i) = (Q_i, \Sigma_i, \xrightarrow[B_i]{})$ their semantics, for all $i = 1, n$. The semantics of C is the labelled transition system $\text{LTS}(C) = (Q_C, \Sigma_C, \xrightarrow[C]{})$ where the set of states $Q_C = \otimes_{i=1}^{n} Q_i$, the set of labels $\Sigma_C = \gamma$ and the set of labelled transitions $\xrightarrow[C]{}$ is defined by the rule:*

$$\textsc{Comp}\ \frac{\begin{array}{c} a = (\{p_i\}_{i\in I}, G_a, F_a) \in \gamma \qquad G_a(\{\mathbf{x}_{p_i}\}_{i\in I}) \qquad \{\mathbf{x}''_{p_i}\}_{i\in I} = F_a(\{\mathbf{x}_{p_i}\}_{i\in I}) \\ \forall i \in I.\ (\ell_i, \mathbf{x}_i) \xrightarrow[B_i]{p_i(\mathbf{x}''_{p_i})} (\ell'_i, \mathbf{x}'_i) \qquad \forall i \notin I.\ (\ell_i, \mathbf{x}_i) = (\ell'_i, \mathbf{x}'_i) \end{array}}{((\ell_1, \mathbf{x}_1), \ldots, (\ell_n, \mathbf{x}_n)) \xrightarrow[C]{a} ((\ell'_1, \mathbf{x}'_1), \ldots, (\ell'_n, \mathbf{x}'_n))}$$

For each $i \in I$, $\mathbf{x}_{p_i}$ above denotes the valuation $\mathbf{x}_i$ restricted to variables of X_{p_i}.

The rule expresses that a composite component $C = \gamma(B_1, \ldots, B_n)$ can execute an interaction $a \in \gamma$ *enabled* in state $((\ell_1, \mathbf{x}_1), \ldots, (\ell_n, \mathbf{x}_n))$, iff (1) for each $p_i \in P_a$, the corresponding atomic component B_i can execute a transition labelled by p_i, and (2) the guard G_a of the interaction holds on the current valuation of variables exported on ports participating in a. Execution of interaction a triggers first the update function F_a which modifies variables exported by ports $p_i \in P_a$. The new values obtained, encoded in the valuation $\mathbf{x}''_{p_i}$, are then used by the components' transitions. The states of components that do not participate in the interaction remain unchanged.

Any finite sequences of interactions $w = a_1 ... a_k \in \gamma^*$ executable by the composite component starting at some given initial state q_0 is named a trace. The set of all traces w from state q_0 is denoted by $\text{TRACES}(C, q_0)$.

3 Information Flow Security

We explore information flow policies [DD77,BLP76,GM82] with focus on the non-interference property. In order to track information we adopt the classification

technique and we define a classification policy where we annotate the information by assigning security levels to different parts of *secBIP* model (data variables, ports and interactions). The policy describes how information can flow from one classification with respect to the other.

As an example, we can classify public information as a *Low* (L) security level and secret (confidential) information as *High* (H) security level. Intuitively *High* security level is more restrictive than *Low* security level and we denote it by $L \subseteq H$. In general, security levels are elements of a security domain, defined as follows:

Definition 6 (security domain). *A security domain is a lattice of the form* $\langle S, \subseteq, \cup, \cap \rangle$ *where:*

- S *is a finite set of security levels.*
- $\subseteq$ *is a partial order "can flow to" on* S *that indicates that information can flow from one security level to an equal or a more restrictive one.*
- $\cup$ *is a "join" operator for any two levels in* S *and that represents the upper bound of them.*
- $\cap$ *is a "meet" operator for any two levels in* S *and that represents the lower bound of them.*

As an example, consider the set $S = \{L, M_1, M_2, H\}$ of security levels that are governed by the "can flow to" relation $L \subseteq M_1$, $L \subseteq M_2$, $M_1 \subseteq H$ and $M_2 \subseteq H$. M_1 and M_2 are incomparable and we note $M_1 \nsubseteq M_2$ and $M_1 \not\supseteq M_2$. This security domain is graphically illustrated in Figure 3.

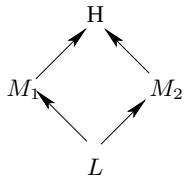

Fig. 3. Example of security domain

Let $C = \gamma(B_1, \ldots B_n)$ be a composite component, fixed. Let X (resp. P) be the set of all variables (resp. ports) defined in all atomic components $(B_i)_{i=1,n}$.

Let $\langle S, \subseteq, \cup, \cap \rangle$ be a security domain, fixed.

Definition 7 (security assignment). *A security assignment for component* C *is a mapping* $\sigma : X \cup P \cup \gamma \to S$ *that associates security levels to variables, ports and interactions such that, moreover, the security levels of ports matches the security levels of interactions, that is, for all* $a \in \gamma$ *and for all* $p \in P$ *it holds* $\sigma(p) = \sigma(a)$.

In atomic components, the security levels considered for ports and variables allow to track intra-component information flows and control the intermediate computation steps. Moreover, inter-components communication, that is, interactions with data exchange, are tracked by the security levels assigned to interactions.

In order to formally introduce the two notions of non-interference for *secBIP* models we need few additional notations, as follows. Let σ be a security assignment for C, fixed.

For a security level $s \in S$, we define $\gamma{\downarrow_s^\sigma}$ the restriction of γ to interactions with security level at most s that is formally, $\gamma{\downarrow_s^\sigma} = \{a \in \gamma \mid \sigma(a) \subseteq s\}$.

For a security level $s \in S$, we define $w|_s^\sigma$ the projection of a trace $w \in \gamma^*$ to interactions with security level lower or equal to s. Formally, the projection is recursively defined on traces as $\epsilon|_s^\sigma = \epsilon$, $(aw)|_s^\sigma = a(w|_s^\sigma)$ if $\sigma(a) \subseteq s$ and $(aw)|_s^\sigma = w|_s^\sigma$ if $\sigma(a) \not\subseteq s$. The projection operator $|_s^\sigma$ is naturally lifted to sets of traces W by taking $W|_s^\sigma = \{w|_s^\sigma \mid w \in W\}$.

For a security level $s \in S$, we define the equivalence $\approx_s^\sigma$ on states of C. Two states q_1, q_2 are equivalent, denoted by $q_1 \approx_s^\sigma q_2$ iff (1) they coincide on variables having security levels at most s and (2) they coincide on control locations having outgoing transitions labeled with ports with security level at most s.

We are now ready to define the two notions of non-interference.

Definition 8 (event non-interference). *The security assignment σ ensures event non-interference of $\gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\forall q_0 \in Q_C^0 : \textsc{traces}(\gamma(B_1, \ldots, B_n), q_0)|_s^\sigma = \textsc{traces}((\gamma{\downarrow_s^\sigma})(B_1, \ldots, B_n), q_0)$$

Event non-interference ensures isolation/security at interaction level. The definition excludes the possibility to gain any relevant information about the occurrences of interactions (events) with strictly greater (or incomparable) levels than s, from the exclusive observation of occurrences of interactions with levels lower or equal to s. That is, an external observer is not able to distinguish between the case where such higher interactions are not observable on execution traces and the case these interactions have been actually statically removed from the composition. This definition is very close to Rushby's [Rus92] definition for transitive non-interference. But, let us remark that event non-interference is not concerned about the protection of data.

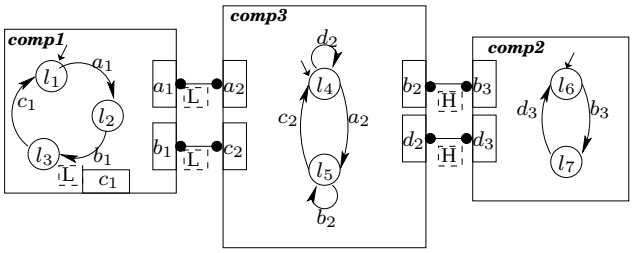

Fig. 4. Example for event non-interference

Example 1. Figure 4 presents a simple illustrative example for event non-interference. The model consists of three atomic components $comp_{i,i=1,2,3}$. Different security levels have been assigned to ports and interactions: $comp_1$ is a low security component, $comp_2$ is a high security component, and $comp_3$ is mixed security component. The security levels are represented by dashed squares related to interactions, internal ports and variables. As a convention, we apply high (H) level for secret data and interactions and low(L) level for public ones. The set of traces is represented by the automaton in Figure 5 (a). The set of projected execution traces at security level L is represented by the automaton depicted in Figure 5 (b). This set is equal to the set of traces obtained by restricted composition, that is, using interaction with security level at most L and depicted in Figure 5 (c). Therefore, this example satisfies the event non-interference condition at level L.

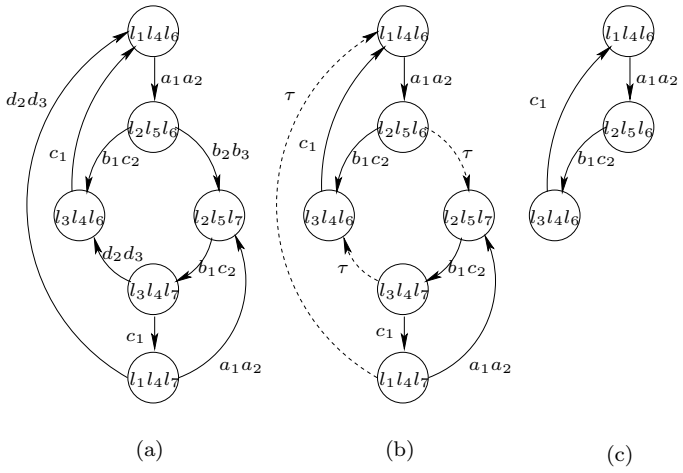

Fig. 5. Sets of traces represented as automata

Definition 9 (data non-interference). *The security assignment σ ensures data non-interference of $C = \gamma(B_1, \ldots, B_n)$ at security level s iff,*

$$\begin{array}{l} \forall q_1, q_2 \in Q_C^0 : \ q_1 \approx_s^\sigma q_2 \Rightarrow \\ \quad \forall w_1 \in \text{TRACES}(C, q_1), w_2 \in \text{TRACES}(C, q_2) : \ w_1|_s^\sigma = w_2|_s^\sigma \Rightarrow \\ \quad\quad \forall q_1', q_2' \in Q_C : \ q_1 \xrightarrow[C]{w_1} q_1' \wedge q_2 \xrightarrow[C]{w_2} q_2' \Rightarrow q_1' \approx_s^\sigma q_2' \end{array}$$

Data non-interference provides isolation/security at data level. The definition ensures that, all states reached from initially indistinguishable states at security level s, by execution of arbitrary but identical traces whenever projected at level s, are also indistinguishable at level s. That means that observation of all

variables and interactions with level s or lower excludes any gain of relevant information about variables at higher (or incomparable) level than s. Compared to event non-interference, data non-interference is a stronger property that considers the system's global states (local states and valuation of variables) and focus on their equivalence along identical execution traces (at some security level).

Example 2. Figure 6 presents an extension with data variables of the previous example from Figure 4. We consider the following two traces $w_1 = \langle a_1a_2, b_2b_3, c_2b_1, d_2d_3, c_1, a_2a_1\rangle$ and $w_2 = \langle a_1a_2, b_2b_3, c_2b_1, c_1, a_2a_1\rangle$ that start from the initial state $((l_1, u = 0, v = 0), (l_4, x = 0, y = 0), (l_6, z = 0, w = 0))$. Although the projected traces at level L are equal, that is, $w_1|^{\sigma}_{L} = w_2|^{\sigma}_{L} = \langle a_1a_2, c_2b_1, c_1, a_1a_2\rangle$, the reached states by w_1 and w_2 are different, respectively $((l_2, u = 4, v = 2), (l_5, x = 3, y = 2), (l_6, z = 1, w = 1))$ and $((l_2, u = 4, v = 2), (l_5, x = 2, y = 2), (l_7, z = 1, w = 0))$ and moreover non-equivalent at low level L. Hence, this example is not data non-interferent at level L.

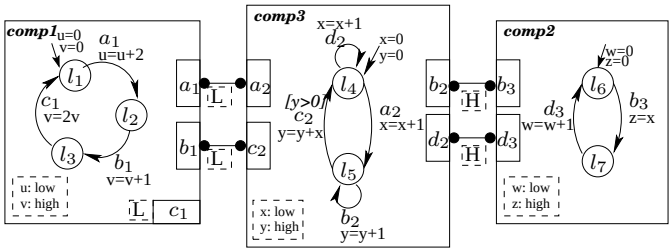

Fig. 6. Example for data non-interference

Definition 10 (secure component). *A security assignment σ is secure for a component $\gamma(B_1, \ldots, B_n)$ iff it ensures both event and data non-interference, at all security levels $s \in S$.*

4 Automated Verification of Non-Interference

We propose hereafter an automated verification technique of non-interference for *secBIP* models based on the so-called unwinding conditions. These conditions were first introduced by Goguen and Meseguer for the verification of transitive non-interference for deterministic systems [GM82]. In general, the unwinding approach reduces the verification of information flow security to the existence of certain unwinding relation. This relation is usually an equivalence relation on system states that respects some additional properties on atomic execution steps, which are shown sufficient to imply non-interference. In the case of *secBIP*, the

additional properties are formulated in terms of individual interactions/events and therefore easier to handle.

Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment for C.

Definition 11 (unwinding relation). *An equivalence $\sim_s$ on states of C is called an unwinding relation for σ at security level s iff the two following conditions hold:*

1. *local consistency*
   $\forall q, q' \in Q_C : \ \forall a \in \gamma : \ q \xrightarrow[C]{a} q' \Rightarrow \sigma(a) \subseteq s \vee q \sim_s q'$
2. *output and step consistency*
   $\forall q_1, q_2, q_1' \in Q_C : \ \forall a \in \gamma :$
   $q_1 \sim_s q_2 \wedge q_1 \xrightarrow[C]{a} q_1' \wedge \sigma(a) \subseteq s \Rightarrow$
   $\exists q_2' \in Q_C : \ q_2 \xrightarrow[C]{a} q_2' \wedge$
   $\forall q_2' \in Q_C : \ q_2 \xrightarrow[C]{a} q_2' \Rightarrow q_1' \sim_s q_2'$

The existence of unwinding relations is tightly related to non-interference. The following two theorems formalize this relation for the two types of non-interference defined. Let C be a composite component and σ a security assignment.

Theorem 1 (event non-interference). *If an unwinding relation $\sim_s$ exists for the security assignment σ at security level s, then σ ensures event non-interference of C at level s.*

Theorem 2 (data non-interference). *If the equivalence relation $\approx_s^\sigma$ is also an unwinding relation for the security assignment σ at security level s, then σ ensures data non-interference of C at level s.*

The two theorems above are used to derive a practical verification method of non-interference using unwinding. We provide hereafter sufficient syntactic conditions ensuring that indeed the unwinding relations $\sim_s$ and $\approx_s$ exist on the system states. These conditions aim to effectively reduce the verification of non-interference to the checking on local constraints on both transitions (intra-component conditions) and interactions (inter-component conditions). Especially, they give an direct way to automate the verification.

Definition 12 (security conditions). *Let $C = \gamma(B_1, \ldots, B_n)$ be a composite component and let σ be a security assignment. We say that C satisfies the security conditions for security assignment σ iff:*

(i) the security assignment of ports, in every atomic component B_i is locally consistent, that is:

- *for every pair of causal transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_2 \xrightarrow{p_2} \ell_3 \ \Rightarrow \ \ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

- *for every pair of conflicting transitions:*

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p_1} \ell_2, \ \tau_2 = \ell_1 \xrightarrow{p_2} \ell_3 \ \Rightarrow \\ \ell_1 \neq \ell_2 \ \Rightarrow \ \sigma(p_1) \subseteq \sigma(p_2)$$

(ii) all assignments $x := e$ occurring in transitions within atomic components and interactions are sequential consistent, in the classical sense:

$$\forall y \in use(e) : \ \sigma(y) \subseteq \sigma(x)$$

(iii) variables are consistently used and assigned in transitions and interactions, that is,

$$\forall \tau \in \cup_{i=1}^{n} T_i \ \ \forall x, y \in X \ : x \in def(f_\tau), y \in use(g_\tau) \ \Rightarrow \\ \sigma(y) \subseteq \sigma(p_\tau) \subseteq \sigma(x) \\ \forall a \in \gamma \ \ \forall x, y \in X \ : x \in def(F_a), y \in use(G_a) \ \Rightarrow \\ \sigma(y) \subseteq \sigma(a) \subseteq \sigma(x)$$

(iv) all atomic components B_i are port deterministic:

$$\forall \tau_1, \tau_2 \in T_i : \ \tau_1 = \ell_1 \xrightarrow{p} \ell_2, \tau_2 = \ell_1 \xrightarrow{p} \ell_3 \ \Rightarrow \\ (g_{\tau_1} \wedge g_{\tau_2}) \text{ is unsatisfiable}$$

The first family of conditions (i) is similar to Accorsi's conditions [AL12] for excluding causal and conflicting places for Petri net transitions having different security levels. Similar conditions have been considered in [FG01,FGF09] and lead to more specific definitions of non-interferences and bisimulations on annotated Petri nets. The second condition (*ii*) represents the classical condition needed to avoid information leakage in sequential assignments. The third condition (*iii*) tackles covert channels issues. Indeed, (*iii*) enforces the security levels of the data flows which have to be consistent with security levels of the ports or interactions (e.g., no low level data has to be updated on a high level port or interaction). Such that, observations of public data would not reveal any secret information. Finally, conditions (*iv*) enforces deterministic behavior on atomic components.

The relation between the syntactic security conditions and the unwinding relations is precisely captured by the following theorem.

Theorem 3 (unwinding theorem). *Whenever the security conditions hold, the equivalence relation $\approx_s^\sigma$ is an unwinding relation for the security assignment σ, at all security level s.*

The following result is the immediate consequence of theorems 1, 2 and 3.

Corollary 1. *Whenever the security conditions hold, the security assignment σ is secure for the component C.*

5 Case study: Web Service Reservation System

We illustrate the *secBIP* framework to handle information flow security issues for a classical example, the web service reservation system proposed in [HV06]. A businessman, living in France, plans to go to Berlin for a private and secret mission. To organize his travel, he uses an intelligent web service who contacts two travel agencies: The first agency, *AgencyA*, arranges flights in Europe and the second agency, *AgencyB*, arranges flights exclusively to Germany. The reservation service obtains in return specific flight information and their corresponding prices and chooses the flight that is more convenient for him.

In this example, there are two types of interference that can occur, (1) data-interference since learning the flight price may reveal the flight destination and (2) event interference, since observing the interaction with *AgencyB* can reveal the destination as well. Thus, to keep the mission private, the flight prices and interactions with *AgencyB* have to be kept confidential.

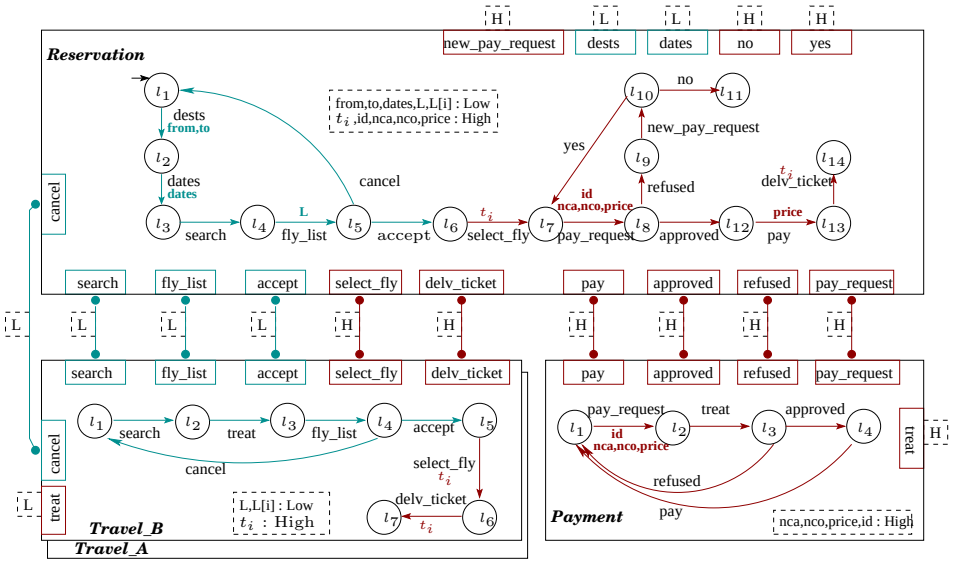

Fig. 7. Model of Reservation Web Service in *secBIP*

The modeling of the system using *secBIP* involves two main distinct steps: first, functional requirements modeling reflecting the system behavior, and second, security annotations enforcing the desired security policy. The model of the system has four components denoted: *Travel_A* and *Travel_B* who are instances from the same component and correspond respectively to AgencyA and AgencyB, and components *Reservation* and *Payment*. To avoid Figure 7 cluttering, we did not represent the interactions with *Travel_A* component. Search parame-

ters are supplied by a user through the *Reservation* component ports *dests* and *dates* to which we associate respectively variables ($from$, to) and $dates$. Next, through search interaction, *Reservation* component contacts *Travel_B* component to search for available flights and obtains in return a list L of specific flights with their corresponding prices. Thereafter, *Reservation* component selects a ticket t_i from the list L and requests the *Payment* component to perform the payment.

All the search parameters $from$, to, $dates$, as well as the flights list L are set to low since users are not identified while sending these queries. Other sensitive data like the selected flight t_i, the price variable p and the payment parameters (identity id, credit card variable cna and code number cno) are set to high. Internal ports *dests* and *dates* as well as *search*, *fly_list*, *accept* interactions are set to low since these interactions (events) do not reveal any information about the client private trip. However, the $select\_fly$ interaction must be set to high since the observation of the selection event from *AgencyB* allow to deduce deduce the client destination. In the case of a selected flight from *AgencyA*, the $select\_fly$ interaction could be set to low since, in this case, the destination could not be deduced just from the event occurrence.

We recall that any system can be proven non-interferent iff it satisfies the syntactic security conditions from Definition 12. Indeed, these conditions hold for the system model depicted in Figure 7. In particular, it can be easily checked that all assignments occurring in transitions within atomic component as well as within interactions are sequential consistent. For example, at the $select\_fly$ interaction we assign a low level security item from the flight list L to a high security level variable ti, formally $t_i = L[i]$. Besides, the security levels assignments to ports exclude inconsistencies due to causal and conflicting transitions, in all atomic components.

6 Related Work

Non-interference properties have been already studied using different model-based approaches. Recently, [SS12] adapted an MDS method for handling information flow security using UML sequence diagrams. Additionally, Petri-nets have been extensively used for system modeling and information flow security verifications tools such as InDico [AWD11] have been developed. A component-based model has been proposed in [ASRL11] and used to study implementation issues of secure information flows. Our presented work on *secBIP* is however different and original in several respects.

First, *secBIP* is a formal framework. Unlike UML, system's runtime behavior is always meaningfully defined and can be formally analyzed. Moreover, *secBIP* provides a system construction methodology for complex systems. Indeed, big systems are functionally decomposed into multiple sub-components communicating through well-defined interactions. Such a structural decomposition of the system is usually not available on Petri-nets models.

Second, *secBIP* handles both event and data-flow non-interference, in a single semantic model. To the best of our knowledge, these properties have never been

jointly considered for component-based models. Nevertheless, the need to consider together event and data flow non-interference has been recently identified in the existing literature. The bottom line is that preserving the safety of data flow in a system does not necessarily preserve safe observability on system's public behavior (i.e., secret/private executions may have an observable impact on system public events). The issue has been recently considered in [AL12], for data leaks and information leaks in business processes based on system's data-flows and work-flows. Also, [BBMP08] showed that formal verification of the system's event behavior is not sufficient to guarantee specific data properties. Furthermore, [FRS05] attempted to fill the gap between respectively language-based and process calculus-based information security and make an explicit distinction between preventing the data leakage through the execution of programs and preventing secret events from being revealed in inter-process communications.

Third, compared to security-typed programming languages [jif,ZZNM02] and operating systems [KYB+07,ZBWM08,EKV+05] enforcing information flow control, *secBIP* is a component-based modeling approach where non-interference is established at a more abstract level. Thus, *secBIP* can be apriori implemented using different programming languages and is independent from a specific operating system and execution platform.

Finally, it is worth mentioning that a lot of classical approaches fall short to handle information flow security [Zda04] for real systems. For *secBIP* we privilege a very pragmatic approach and provide simple (syntactic) sufficient conditions allowing to automate the verification of non-interference. These conditions allow to eliminate a significant amount of security leakages, especially covert channels, independently from system language or the execution platform. However, these conditions can be very restrictive in some cases and a system designer may be interested to relax the non-interference properties.

7 Conclusion and Future Work

We present a MDS framework to secure component-based systems. We formally define two types of non-interference, respectively event and data non-interference. We provide a set of sufficient syntactic conditions which simplify verification of non-interference. These conditions are extensions of security typed language rules applied to our model. The use of our framework has been demonstrated to secure a web service application.

This work is currently being extended in two directions. First, we are investigating additional security conditions allowing to relax the non-interference property and control where downgrading can occur. Second, we are working towards the implementation of a complete design flow for secure systems based on *secBIP*. As a first step, we shall implement the verification method presented for annotated *secBIP* models. Then, use these models for generation of secure implementations, that is, executable code where the security properties are enforced by construction, at the generation time.

References

[AL12] Rafael Accorsi and Andreas Lehmann. Automatic information flow analysis of business process models. In *10th International Conference on Business Process Management (BPM'12)*, volume 7481 of *LNCS*, pages 172–187. Springer, 2012.

[ASRL11] Takoua Abdellatif, Lilia Sfaxi, Riadh Robbana, and Yassine Lakhnech. Automating information flow control in component-based distributed systems. In *14th International ACM Sigsoft Symposium on Component Based Software Engineering (CBSE'11)*, pages 73–82. ACM, 2011.

[AWD11] Rafael Accorsi, Claus Wonnemann, and Sebastian Dochow. Swat: A security workflow analysis toolkit for reliably secure process-aware information systems. In *Sixth International Conference on Availability, Reliability and Security (ARES'11)*, pages 692–697. IEEE, 2011.

[BBB+11] Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. Rigorous component-based design using the BIP framework. *IEEE Software, Special Edition – Software Components beyond Programming – from Routines to Services*, 28(3):41–48, 2011.

[BBMP08] Cesare Bartolini, Antonia Bertolino, Eda Marchetti, and Ioannis Parissis. *Architecting Dependable Systems V*, chapter Data Flow-Based Validation of Web Services Compositions: Perspectives and Examples, pages 298–325. Springer, 2008.

[BBS06] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)*, pages 3–12. IEEE Computer Society Press, 2006.

[BDL06] David Basin, Jürgen Doser, and Torsten Lodderstedt. Model driven security: from uml models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15:39–91, 2006.

[BLP76] E. D. Bell and J. L. La Padula. Secure computer system: Unified exposition and Multics interpretation, 1976.

[DD77] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, pages 504–513, 1977.

[EKV+05] Petros Efstathopoulos, Maxwell Krohn, Steve VanDeBogart, Cliff Frey, David Ziegler, Eddie Kohler, David Mazières, Frans Kaashoek, and Robert Morris. Labels and Event Processes in the Asbestos Operating System. *SIGOPS Operating Systems Review*, 39(5):17–30, 2005.

[FG01] Riccardo Focardi and Roberto Gorrieri. Classification of Security Properties (Part I: Information Flow). In *Revised lectures of IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design (FOSAD'00)*, volume 2171 of *LNCS*, pages 331–396. Springer, 2001.

[FGF09] Simone Frau, Roberto Gorrieri, and Carlo Ferigato. Petri net security checker: Structural non-interference at work. In *Formal Aspects in Security and Trust, 5th International Workshop (FAST'08), Revised Lectures*, volume 5491 of *LNCS*, pages 210–225. Springer, 2009.

[FRS05] Riccardo Focardi, Sabina Rossi, and Andrei Sabelfeld. Bridging language-based and process calculi security. In *Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *LNCS*, pages 299–315. Springer, 2005.

[GM82] Joseph A. Goguen and José Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[HV06] Dieter Hutter and Melanie Volkamer. Information flow control to secure dynamic web service composition. In *Security in Pervasive Computing (SPC'06)*, volume 3934 of *LNCS*, pages 196–210. Springer, 2006.

[jif] http://www.cs.cornell.edu/jif/.

[Kuh98] D. Richard Kuhn. Role Based Access Control on MLS Systems without Kernel Changes. In *ACM Workshop on Role Based Access Control (RBAC'98)*, pages 25–32. ACM, 1998.

[KYB+07] Maxwell Krohn, Alexander Yip, Micah Brodsky, Natan Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Morris. Information Flow Control for Standard OS Abstractions. *SIGOPS Operating Systems Review*, 41(6):321–334, 2007.

[Man00] Heiko Mantel. Possibilistic Definitions of Security - An Assembly Kit. In *13th IEEE Workshop on Computer Security Foundations (CSFW'00)*, page 185. IEEE Computer Society, 2000.

[McC88] Daryl McCullough. Noninterference and the composability of security properties. In *Security and Privacy (SP'88)*, pages 177–186. IEEE Computer Society, 1988.

[McL94] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Security and Privacy (SP'94)*, page 79. IEEE Computer Society, 1994.

[Rus92] John Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL-92-2, SRI International, 1992.

[SM03] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), 2003.

[SQSL05] Jianjun Shen, Sihan Qing, Qingni Shen, and Liping Li. Covert channel identification founded on information flow analysis. In *Computational Intelligence and Security (CIS'05)*, volume 3802 of *LNCS*, pages 381–387. Springer, 2005.

[SS01] Andrei Sabelfeld and David Sands. A per model of secure information flow in sequential programs. *Higher Order Symbolic Computation*, 14(1):59–91, 2001.

[SS12] Fredrik. Seehusen and Ketil Stølen. *Dependability and Computer Engineering: Concepts for Software-Intensive Systems*, chapter A Method for Model-driven Information Flow Security, pages 199–229. IGI Global, 2012.

[SSM98] Ravi Sandhu, Ravi S, and Qamar Munawer. How to do discretionary access control using roles. In *ACM Workshop on Role-based access control (RBAC'98)*, pages 47–54. ACM, 1998.

[SV98] Geoffrey Smith and Dennis Volpano. Secure information flow in a multi-threaded imperative language. In *Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM, 1998.

[ZBWM08] Nickolai Zeldovich, Silas Boyd-Wickizer, and David Mazières. Securing distributed systems with information flow control. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, pages 293–308. USENIX Association, 2008.

[Zda04] Steve Zdancewic. Challenges for information-flow security. In *Programming Language Interference and Dependence (PLID'04)*, 2004.

[ZL97] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Security and Privacy (SP'97)*, pages 94–102. IEEE Computer Society, 1997.

[ZZNM02] Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, and Andrew C. Myers. Secure program partitioning. *ACM Transactions on Computer Systems*, 20(3):283–328, August 2002.

Appendix

Proof of Theorem 1

Proof. We shall prove $\textsc{traces}(\gamma(B_1,\ldots,B_n),q_0)|_s^\sigma = \textsc{traces}((\gamma\downarrow_s^\sigma)(B_1,\ \ldots,\ B_n),\ q_0)$ by double inclusion. "$\supseteq$" inclusion: Independently of the unwinding relation, by using elementary set properties it holds that $\textsc{traces}((\gamma\downarrow_s^\sigma)(B_1,\ \ldots,\ B_n),q_0) = \textsc{traces}((\gamma\downarrow_s^\sigma)(B_1,\ \ldots,\ B_n),q_0)|_s^\sigma \subseteq \textsc{traces}(\gamma(B_1,\ \ldots,\ B_n),q_0)|_s^\sigma$. "$\subseteq$" inclusion: This direction is an immediate consequence of the following Lemma 1. It states that for every trace w in $\textsc{traces}(\gamma(B_1,\ldots,B_n),q_0)$ its projection $w|_s^\sigma$ is also a valid trace in $\textsc{traces}(\gamma(B_1,\ldots,B_n),q_0)$. But, this also means that $w|_s^\sigma$ is a valid trace in $\textsc{traces}((\gamma\downarrow_s^\sigma)(B_1,\ldots,B_n),q_0)$ which proves the result.

Lemma 1. *In the conditions of Theorem 1, for every trace w in $\textsc{traces}(\gamma(B_1,\ \ldots,\ B_n),q_0)$, for every state q such that $q_0 \xrightarrow[C]{w} q$, the projected trace $w|_s^\sigma$ is also a valid trace in $\textsc{traces}(\gamma(B_1,\ldots,B_n),q_0)$ and moreover, for every state q' such that $q_0 \xrightarrow[C]{w|_s^\sigma} q'$ it holds $q \sim_s q'$.*

Proof. The lemma is proved by induction on the length of the trace w. For the empty trace $w=\epsilon$ verification is trivial: $\sim_s$ holds for the initial state $q_0 \sim_s q_0$ and $\epsilon = \epsilon|_s^\sigma$. By induction hypothesis, let assume the property holds for traces of length n. We shall prove the property for traces of length $n+1$. Let $w' = wa$ be an arbitrary trace of length $n+1$, let w be its prefix (trace) of length n and let a be the last interaction. Consider states q, q_1 such that $q_0 \xrightarrow[C]{w} q \xrightarrow[C]{a} q_1$. By the induction hypothesis we know that $w|_s^\sigma$ is a valid trace and for all states q' such that $q_0 \xrightarrow[C]{w|_s^\sigma} q'$ it holds $q \sim_s q'$. We distinguish two cases, depending on the security level of a:

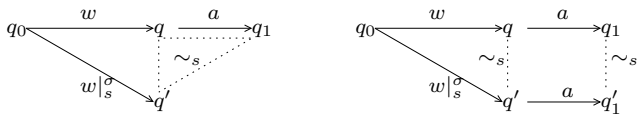

Fig. 8. Proof illustration for Lemma 1

– $\sigma(a) \not\subseteq s$: In this case, $w'|_s^\sigma = w|_s^\sigma$ hence, $w'|_s^\sigma$ is a valid trace as well, reaching the same states q'. Moreover, since a is invisible for s, the unwinding condition (1) ensures that $q \sim_s q_1$. By transitivity, this implies that $q_1 \sim_s q'$, which proves the result.
– $\sigma(a) \subseteq s$: In this case, $w'|_s^\sigma = w|_s^\sigma a$. From the unwinding condition (2), since $q \sim_s q'$ and a is visible and enabled in q then, a must also be enabled in q'. Therefore, $w|_s^\sigma$ can be extended with a from state q' to some q_1' hence, $w'|_s^\sigma$ is indeed a valid trace. Moreover, since $q \sim_s q'$ the unwinding condition (2) ensures also that $q_1 \sim_s q_1'$, which proves the result.

Proof of Theorem 2

Proof. Let us consider two equivalent states $q_1 \approx_s^\sigma q_2$.
The *first condition* for data non-interference requires that, for any trace w_1 from q_1 there exists a trace w_2 from q_2 having the same projection at level s, that is, $w_1|_s^\sigma = w_2|_s^\sigma$.

We shall prove a slightly stronger property, namely, the trace w_2 can be chosen such that, the successors q_1' and q_2' of respectively q_1 by w_1 and q_2 by w_2 are moreover equivalent, that is, $q_1' \approx_s^\sigma q_2'$. The proof is by induction on the length of the trace w_1. *The base case*: for the empty trace $w_1 = \epsilon$ we take equally $w_2 = \epsilon$ we immediately have $q_1' = q_1 \approx_s^\sigma q_2 = q_2'$. *The induction step:* we assume, by induction hypothesis that the property holds for all traces w_1 such that $|w_1| \leq n$ and we shall prove it for all traces w_1' such that $|w_1'| = n + 1$. Let a be the last interaction executed in w_1', that is, $w_1' = w_1 a$ with $|w_1| = n$. Let q_1'' be the state reached from q_1 by w_1. From the induction hypothesis, there exists a trace w_2 that leads q_2 into q_2'' such that $w_1|_s^\sigma = w_2|_s^\sigma$ and moreover $q_1'' \approx_s^\sigma q_2''$. We distinguish two cases, depending on the security level of a:

– $\sigma(a) \not\subseteq s$: since $\approx_s^\sigma$ is unwinding and $q_1'' \xrightarrow[C]{a} q_1'$ it follows that $q_1'' \approx_s^\sigma q_1'$. In this case, we take $w_2' = w_2$ and $q_2' = q_2''$ which ensures both $w_1'|_s^\sigma = w_1|_s^\sigma = w_2|_s^\sigma = w_2'|_s^\sigma$ and $q_1' \approx_s^\sigma q_1'' \approx q_2'' = q_2'$.
– $\sigma(a) \subseteq s$: since $\approx_s^\sigma$ is unwinding and $q_1'' \approx_s^\sigma q_2''$ and $q_1'' \xrightarrow[C]{a} q_1'$ there must exists q_2' such that $q_2'' \xrightarrow[C]{a} q_2'$ and moreover, for any such choice $q_1' \approx_s^\sigma q_2'$. Hence, in this case, the trace $w_2' = w_2 a$ executed from q_2 and leading to q_2' satisfies our property, namely $w_1'|_s^\sigma = w_1|_s^\sigma a = w_2|_s^\sigma a = w_2'|_s^\sigma$ and $q_1' \approx_s^\sigma q_2'$.

The *second condition* for data non-interference requires that, for any traces w_1 and w_2 with equal projection on security level s, that is $w_1|_s^\sigma = w_2|_s^\sigma$, any successor states q_1' and q_2' of respectively q_1 by w_1 and q_2 by w_2 are also equivalent at level s. This property is proved also by induction on $|w_1| + |w_2|$, that is, on the sum of the lengths of traces w_1, w_2. *The base case*: for empty traces $w_1 = w_2 = \epsilon$ we have that $q_1' = q_1$ and $q_2' = q_2$ and hence trivially $q_1' \approx_s^\sigma q_2'$. *The induction step*: we assume, by induction hypothesis that the property holds for any traces w_1, w_2 such that $|w_1| + |w_2| \leq n$ and we shall prove it for all traces w_1', w_2' such that $|w_1'| + |w_2'| = n + 1$. We distinguish two cases, depending on the security levels of the last interactions occurring in w_1' and w_2'.

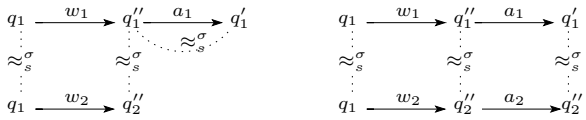

Fig. 9. Proof illustration for Theorem 2

- at least one of the last interactions in w_1' or w_2' has a security level not lower or equal to s. W.l.o.g, consider that indeed $w_1' = w_1 a_1$ and $\sigma(a_1) \not\subseteq s$. This situation is depicted in Figure 9, (left).
  Let q_1'' be the state reached from q_1 after w_1. Since $w_1'|_s^\sigma = w_2'|_s^\sigma$ and $\sigma(a_1) \not\subseteq s$ it follows that $w_1|_s^\sigma = w_1'|_s^\sigma = w_2'|_s^\sigma$. The induction hypothesis holds then for w_1 and w_2' because $|w_1| + |w_2'| = n - 1$ and hence we have that $q_1'' \approx_s^\sigma q_2'$. Moreover, q_1' is a successor of q_1'' by interaction a_1. Since the security level of a_1 is not lower or equal to s, and $\approx_s^\sigma$ is an unwinding relation at level s, it follows from the local consistency condition that $q_1'' \approx_s^\sigma q_1$. Then, by transitivity of $\approx_s^\sigma$ we obtain that $q_1' \approx_s^\sigma q_2'$.
- the last interactions of both traces w_1' and w_2' have security level lower or equal to s. That is, consider $w_1' = w_1 a_1$ and $w_2' = w_2 a_2$ with $\sigma(a_1) \subseteq s$, $\sigma(a_2) \subseteq s$. This situation is depicted in Figure 9, (right).
  Let q_1'' and q_2'' be the states reached respectively from q_1 by w_1 and from q_2 by w_2. Since $\sigma(a_1) \subseteq s$,$\sigma(a_2) \subseteq s$ we have $w_1'|_s^\sigma = w_1|_s^\sigma a_1$, $w_2'|_s^\sigma = w_2|_s^\sigma a_2$. From the hypothesis, $w_1'|_s^\sigma = w_2'|_s^\sigma$, it follows that both $a_1 = a_2$ and $w_1|_s^\sigma = w_2|_s^\sigma$. Therefore, the induction hypothesis can be applied for traces w_1, w_2 because $|w_1| + |w_2| = n - 2$ and hence, we obtain $q_1'' \approx_s^\sigma q_2''$. But now, q_1' and q_2' are immediate successors of two equivalent states q_1'' and q_2'' by executing some interaction $a = a_1 = a_2$, having security level lower or equal to s. Since, $\approx_s^\sigma$ is an unwinding relation at level s, it follows from the step consistency condition that successors states q_1' and q_2' are also equivalent at level s, hence, $q_1' \approx_s^\sigma q_2'$.

Proof of Theorem 3

Proof. Let s be an arbitrary fixed security level. We shall prove that $\approx_s^\sigma$ satisfies the local, output and step consistency, as required by Definition 11.

local consistency: Let $q, q' \in Q_C$ be two states such that $q \xrightarrow[C]{a} q'$. We must show that if $\sigma(a) \not\subseteq s$ then $q \approx_s^\sigma q'$.

All variables x modified by a itself and by the transitions participating in a are such that $\sigma(a) \subseteq \sigma(x)$ (security conditions, (*iii*)). Then, since $\sigma(a) \not\subseteq s$, it also follows that all variables modified have security level greater or incomparable to s. Conversely, it follows that all variables with security levels lower or equal to s are not modified by a, hence they have the same values in q in q'.

Regarding control locations, we proceed by contradiction. Let consider that some component B_i is respectively at ℓ_i in q and at ℓ_i' in q' and moreover, either

at ℓ_i or ℓ'_i there exists transitions with ports having security levels lower or equal s. Since the location of B_i has changed, it means that it has participated in the interaction a using some transition $\tau_i = \ell_i \xrightarrow{p_i} \ell'_i$. Let consider the two situations:

- there exists transitions with security level lower or equal to s at ℓ'_i. Let $\tau'_i = \ell'_i \xrightarrow{p'_i} \ell''_i$ such a transition. This situation contradicts the security conditions (i), as τ'_i is causally dependent on τ_i and has a different, yet not increased security level i.e., $\sigma(p_i) = \sigma(a) \not\subseteq s$ and $\sigma(p'_i) \subseteq s$.
- there exists transitions with security level lower or equal to s at ℓ_i. Let $\tau'_i = \ell_i \xrightarrow{p'_i} \ell''_i$ such a transition. This situation contradicts again the security conditions (i): as τ_i and τ'_i are now conflicting it must be $\sigma(p_i) \subseteq \sigma(p'_i)$ which contradicts that $\sigma(p_i) = \sigma(a) \not\subseteq s$ and $\sigma(p'_i) \subseteq s$.

Henceforth, as the two situations lead to contradiction we conclude that, either $\ell_i = \ell'_i$, or otherwise, neither in ℓ_i or ℓ'_i there exists transitions with ports having security level lower or equal to s. This conclude the proof of $q \approx^{\sigma}_{s} q'$

output and step consistency: Let q_1, q_2 be two equivalent states $q_1 \approx^{\sigma}_{s} q_2$.

Let a be an interaction with security level lower or equal to s enabled in q_1. We show that the same interaction is enabled in q_2. All components participating in a use transitions with ports with the same level as a, hence at most s. Therefore, these components are at control locations where there are outgoing transitions with level at most s. Then, these components are precisely in the same locations in q_2 since $q_1 \approx^{\sigma}_{s} q_2$.

Moreover, all the guards of the interacting transitions as well as the guard of the interaction use variables with security level lower or equal to $\sigma(a)$ and consequently, lower or equal to s (security conditions, (iii)). But again, $q_1 \approx^{\sigma}_{s} q_2$ implies that all variables with levels lower or equal to s have equal values in q_1 and q_2. Hence, the guards used in a have the same evaluation in q_1 or q_2. Together with equality on control locations, established earlier, this implies that a is enabled in q_2.

Let now consider two arbitrary states q'_1, q'_2 reached by a from respectively q_1 and q_2. We must show that $q'_1 \approx^{\sigma}_{s} q'_2$. First, as $\sigma(a) \subseteq s$, it follows that, as explained before, enabledness of a depends exclusively on identical parts of q_1 and q_2. Moreover, due to security conditions (iv) it follows also that the execution of a synchronizes *exactly* the same set of transitions when executed either from q_1 or from q_2. Hence, in the successor states q'_1 and q'_2 all interacting atomic components have moved towards the same locations. The equality condition on the control locations is therefore satisfied. Furthermore, using security conditions (ii) it holds that all variables modified by transitions involved in a, if they have security values lower or equal to s, they will be assigned the same values. That is, the assigned expression use only variables with a lower security level, and hence identical on q_1 and q_2. This ensures equality of variables with security level lower or equal to s in q'_1 and q'_2, which conclude the proof.